13/Brief
2/13/04

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	:
Baldwin	: Art Unit: 2676
AN 09/591,226	: Examiner: Tung, Kee M.
Filed: 06/09/2000	: Atty's Docket: TD-156

For: AUTONOMOUS ADDRESS TRANSLATION IN GRAPHICS SUBSYSTEM (confirmation no. 3469)

APPEAL BRIEF

Honorable Commissioner of Patents and Trademarks
Alexandria, VA 22313

Sir:

Appellant herewith respectfully submit that the Examiner's decision of 06/04/2003, finally rejecting Claims 1-6 in the present application, should be reversed in view of the following arguments and authorities.

TABLE OF CONTENTS

Real Party in Interest

The real party in interest, and assignee of this case, is 3DLabs Inc., Ltd., which is now a subsidiary of Creative Technologies, of Singapore.

Related Appeals and Interferences

To the best knowledge and belief of the undersigned attorney, there are none. However, please note that 09/133,741 and 09/280,250 are cases of common assignee which are currently on appeal, and which involve the same general field of technology.

Status of Claims

Claims 1-6 are the subject of this appeal. No other claims are pending.

Status of Amendments

An Amendment After Final Rejection filed on December 4, 2003, has been entered (see Advisory Action of December 17, 2003).

Summary of Invention

The following summary refers to disclosed embodiments and their advantages, but does not delimit any of the claimed inventions.

Virtual Texture Memory

Virtualization of texture memory, like virtualization of host memory, gives the user the impression of a memory space which is larger than can be physically accommodated in real memory. This is achieved by partitioning the memory space into a small physical working set and a large virtual set with dynamic swapping between the two. For virtual memory management in CPUs, the physical working set is main memory. and the virtual set is disk storage.[1]

The swapping required for virtual memory management is normally done automatically (as far as the application software is concerned). There is a vast amount of literature concerning CPU-based virtual memory systems and their management.[2]

The apparently-larger virtual texture memory space increases performance as the optimum set of textures (or part of textures) are chosen for residence by the hardware. It also simplifies the management of texture memory by the driver and/or application where either or both try to manage the memory manually. This is akin to program overlays before the days of virtual memory on CPUs where the program had to dynamically load and unload segments of itself.[3]

The present inventor has realized that managing the texture memory in the driver or by the application is very difficult (or impossible) to do properly because:

1. What does the driver/application do when it runs out of memory and needs to fit another texture in? Which texture(s) does it delete?

---

[1] Paragraph beginning on page 9, line 13.

[2] Paragraph beginning on page 9, line 20.

[3] Paragraph beginning on page 9, line 23.

2. The texture has to be completely resident and physically contiguous so a large enough space must be made available.
3. A texture which is about to be used MUST NOT be deleted or moved: otherwise all command buffers will be outdated.
4. In some cases, a texture map will not fit into memory even when all other textures are deleted (a 2Kx2K 32bpp texture map takes 16MBytes of memory).
5. The texture heap must be compacted to reclaim storage.[4]

The idea of applying virtual management techniques to textures in 3D graphics hardware appears to be suggested, for example, by U.S. Patent 5,790,130 to Gannett. This patent suggests that "A graphics hardware device, coupled to the host computer, renders texture mapped images, and includes a local memory that stores at least a portion of the texture data stored in the system memory at any one time. A software daemon runs on the processor of the host computer and manages transferring texture data from the system memory to the local memory when needed by the hardware device to render an image." (Abstract) This and/or other virtual texture memory schemes are believed to have been used in some products of HP and SGI. However, the present inventor has realized that these schemes are ill-suited for most personal computer applications (and many workstation applications). The main aim in these implementation seems to have been to allow very large texture maps (16Mx16M or larger) to be used. By contrast, the innovations in the present application are not motivated only by desire for such large maps, but to remove the software problems in managing the comparatively small amount of texture storage (vs the large amounts of texture storage in SGI and HP machines) efficiently. Thus, it is possible that the architectural innovations disclosed herein can be used in combination with those used by SGI and HP.[5]

---

[4] Paragraph beginning on page 9, line 29.

[5] Paragraph beginning on page 10, line 7.

Autonomous Address Translation In Graphics Subsystem

As noted above, virtual memory architectures have long been used in general-purpose computers. However, there turn out to be some surprising difficulties in using this idea in computer graphics, especially for texture memory. The present application discloses several innovations related to virtualization and caching of texture memory.[6]

According to various inventions claimed in the present application, the texture memory management function can be used to manage texture storage in the host memory in addition to the texture storage in normal texture memory, providing additional capability for optimized management.[7]

Some textures should not be downloaded into storage on the graphics card because they will only be used once. (For example, with some applications, it might be known that the textures will be dynamically updated.) In such cases, the cost of downloading them does not compensate for the faster local access. In this case, the same virtual management mechanisms can be used to allow non-contiguous texture allocation in host memory (but without the download to level-i memory). They can also page from level-3 memory to level-2 memory. This mechanism goes beyond the address mapping functionality built in as part of the AGP protocols (which use a GART table in the core logic chip set to do the logical-to-physical mapping) in that it supports the level-3 memory and is part of an integrated/unified system.[8]

It takes a lot of effort for software to manage memory, and much of this relates to the desire to minimize fragmentation while managing big blocks of texture; so in the past a lot of overhead has been wasted on compacting memory to collect free space. The GART tables have been used to do some logical-to-physical mapping, but were NOT user accessible.[9]

---

[6] Paragraph beginning on page 10, line 24.

[7] Paragraph beginning on page 10, line 29.

[8] Paragraph beginning on page 11, line 1.

[9] Paragraph beginning on page 11, line 11.

Since the presently preferred embodiment provides a user accessible mechanism in place to do logical-to-physical mapping, it can be used in other ways too. For example, the presently preferred embodiment allows "textureexecute," i.e. operation without downloading to local memory.[10]

Note that the preferred controller does not interfere with the fetch to host memory - only the fetch to CARD memory. Note also that the preferred embodiment, unlike AGP, gives the capability to generate interrupts when accessing an AGP texel.[11]

Since the GART table is out of our control, it is preferable not to use it; the alternative mechanism also allows other things to be done, e.g. generating an interrupt to read an AGP texture without downloading it.[12]

This is particularly useful if it is known that only a very few texels will be used, or if the textures are very dynamic, or if a particular texture will only be needed once. In the presently preferred embodiment, there is one "interrupt" bit per page.[13]

Virtual Texture Management

Texture maps can be stored in physical memory or in logical/virtual memory. If the texture map is stored in physical memory, then it must be physically contiguous and present before that texture is used.[14]

The management of physical textures is complicated by the fact that an application can request more textures than can fit into on-card memory so the textures need to be dynamically swapped; however, this is not an easy task to do well because:

---

[10] Paragraph beginning on page 11, line 15.

[11] Paragraph beginning on page 11, line 19.

[12] Paragraph beginning on page 11, line 22.

[13] Paragraph beginning on page 11, line 25.

[14] Paragraph beginning on page 31, line 8.

• The need to swapping and usage are decoupled in time by the DMA buffers.

• The memory granularity is controlled by the texture map size so is continually changing.

• Memory gets fragmented.

• There is no clear replacement policy.

There are a number of possible approaches to solving this problem:

• Increase the amount of physical memory to hold texture maps. This is not always possible due to cost or board area constraints and, in any case, just delays the point at which the problem will re-occur rather than fixing it altogether.

• Allow textures to be executed out of host memory via the AGP or PCI bus. This is a similar solution to the previous one, except it does not have the cost or board area constraints (at least as far as the graphics board is concerned). The downside of this is the bandwidth across the AGP bus is likely to be inferior to the bandwidth out of local memory. Also, the latency for the texture data to arrive may degrade texture performance. This method is supported by setting the HostTexture bit in the TextureMapWidth registers. These texture reads will be done across the AGP bus. The PCI bus can be used, but because it lacks the efficient random in-page addressing AGP has, the texture accesses will be very slow. Note that there may be system reasons why such a method will not work or work poorly. A system with a GLINT Gamma cannot do this type of access (across AGP) and multiple RX's would require too much bandwidth and not interleave accesses very well.

• The final solution is to treat the texture addresses as logical or virtual addresses. The logical part allows texture maps to be stored in non-contiguous physical pages (a page is 4K bytes). This simplifies the memory management aspect as the granularity now is at the page level. The virtual part allows the dynamic paging of textures out of host or system memory with or without any assistance from the host CPU. This is done on demand so borrows many of the techniques used for CPU memory management.

The virtual texture management (of which the logical addressing is a necessary sub-set) is implemented as standard in this unit.[15]

Host textures can also be managed; the main difference is that **no texture data is** downloaded, but is accessed "in situ" using the side band addressing capability of the AGP texture execute mode.[16]

[15] Paragraph beginning on page 31, line 12.

[16] Paragraph beginning on page 33, line 11.

Issues

Issue 1 – Are Claims 1 and 4 obvious over *Peddada et al.* in view of *Porterfield*?

Issue 2 – Are Claims 2, 3, 5, and 6 obvious over *Porterfield* in view of *Mergard et al.* or *Poirion* or *Chen et al*?

Issue 3 – Are Claims 2, 3, 5, and 6 obvious over *Peddada et al.* in view of *Mergard et al.* or *Poirion* or *Chen et al*?

Grouping of Claims

For each ground of rejection which Appellant contests herein which applies to more than one claim, such additional claims, to the extent identified and argued below, do not stand or fall together.

ARGUMENT

Review of the References

Some of the major technical differences between the references applied and the disclosure of the present application will now be reviewed.

Peddada et al.

Peddada et al. (U.S. Patent No. 6,295,068) relates to a 3D graphics driver that manages a texture cache in the local graphics memory. The driver disclosed by *Peddada et al.* does not appear to manage texture memory in the main memory nor does it suggest a software that has user accessible mechanism in place to do logical-to-physical mapping into a main system memory. *Peddada et al.* also does not disclose allowing the graphics accelerator itself direct access to the main memory or any other innovations with regard to graphics accelerators.

Porterfield

Porterfield (U.S. Patent No. 6,249,853) relates to a modular device for storing, addressing, and retrieving graphics data from main memory. The graphics accelerator chip disclosed by *Porterfield* does not contain a memory management function nor is it capable of reading non-contiguous textures directly from said main memory.

Mergard et al.

Mergard et al. (U.S. Patent No. 5,941,968) relates to a computer system in which the graphics controller and system memory are coupled to a high-speed data bus for improved bus concurrency. *Mergard et al.* does not appear to disclose or suggest any innovations with regard to graphics accelerators.

Poirion

Poirion (U.S. Patent No. 6,232,990) relates to a single-chip chipset in which the graphics controller is integrated with the chipset and can access

system memory. *Poirion* does not appear to disclose or suggest any innovations with regard to graphics accelerators.

Chen et al.

Chen et al. (U.S. Patent No. 6,292,201) relates to an integrated circuit device. *Chen et al.* does not appear to disclose or suggest any innovations with regard to graphics accelerators.

Rejections under §103

Legal Standards

Any obviousness rejection requires some showing of motivation to modify or combine the reference(s) applied, in a way that meets the claimed invention. In the long line of case law stemming from *Graham v. John Deere,*[17] many Federal Circuit opinions have summarized this legal requirement; one recent, frequently-cited case is *In re* Rouffet.[18] This opinion states very emphatically (with many citations):[19]

> [T]he examiner must show reasons that the skilled artisan, confronted with the same problems as the inventor and with no knowledge of the claimed invention, would select the elements

---

[17] *Graham v. John Deere Co.*, 383 U.S. 1, 148 USPQ 459 (1966).

[18] 149 F.3d 1350, 47 USPQ2d 1453 (Fed.Cir. 1998).

[19] Cases cited to support the Court's emphasis on motivation include: *In re Geiger*, 815 F.2d 686, 2 USPQ2d 1276 (Fed.Cir. 1987); *Pro-Mold & Tool Co. v. Great Lakes Plastics, Inc.*, 75 F.3d 1568, 37 USPQ2d 1626 (Fed.Cir. 1996); *In re Sernaker*, 702 F.2d 989, 217 USPQ 1 (Fed.Cir. 1983); *Ashland Oil, Inc. v. Delta Resins & Refractories, Inc.*, 776 F.2d 281, 227 USPQ 657 (Fed.Cir. 1985); *In re Beattie*, 974 F.2d 1309, 24 USPQ2d 1040 (Fed.Cir. 1992); *Lindemann Maschinenfabrik GmbH v. American Hoist & Derrick Co.*, 730 F.2d 1452, 221 USPQ 481 (Fed.Cir. 1984); *Environmental Designs, Ltd. v. Union Oil Co.*, 713 F.2d 693, 218 USPQ 865 (Fed.Cir. 1983); *Richdel, Inc. v. Sunspool Corp.*, 714 F.2d 1573, 219 USPQ 8 (Fed.Cir. 1983); and *Sensonics, Inc. v. Aerosonic Corp.*, 81 F.3d 1566, 38 USPQ2d 1551 (Fed.Cir. 1996).

from the cited prior art references for combination **in the manner claimed**.[20]

Even more recently, *In re Lee*[21] emphasized (with many citations[22]) that: "The need for specificity pervades this authority."

A similarly emphatic discussion is found in *Ruiz v. A.B. Chance Co.*, where the Court, after a long review of the case law, concludes that the showing of combinability must be "**clear and particular**."[23]

In *Thrift,*[24] a rejection which "does not discuss the unique limitations" of the claims was held to be "simply inadequate on its face." In this case, a rejection was held "not supported by substantial evidence because **the cited references do not support each limitation** of claim 11." See *In re Vaeck*, 947 F.2d 488, 493, 20 USPQ2d 1438, 1443 (Fed.Cir. 1991)."[25]

---

[20] In re Rouffet, 149 F.3d at 1357 (emphasis added).

[21] 277 F.3d 1338, 61 USPQ2d 1430 (Fed.Cir. 2002), cited and applied, e.g., by *In re Huston*, 308 F.3d 1267, 64 USPQ2d 1801 (Fed.Cir. 2002). Note that Huston upheld a rejection, but explicitly applied *Lee*'s standard.

[22] The Court's opinion cites *In re Dance* 160 F.3d 1339, 48 USPQ2d 1635 (Fed.Cir. 1998), *Rouffet*, and many other cases in the same vein, including *In re Grasselli*, 713 F.2d 731, 218 USPQ 769 (Fed.Cir. 1983); *McGinley v. Franklin Sports, Inc.*, 262 F.3d 1339, 60 USPQ2d 1001 (Fed.Cir. 2001); *Brown & Williamson Tobacco Corp. v. Philip Morris, Inc.*, 229 F.3d 1120, 56 USPQ2d 1456 (Fed.Cir. 2000); *C.R. Bard, Inc., v. M3 Systems, Inc.*, 157 F.3d 1340, 48 USPQ2d 1225 (Fed.Cir. 1998); *In re Dembiczak*, 175 F.3d 994, 50 USPQ2d 1614 (Fed.Cir. 1999); *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed.Cir. 1988); *ACS Hosp. Sys., Inc. v. Montefiore Hosp.*, 732 F.2d 1572, 221 USPQ 929 (Fed.Cir. 1984); and *In re Fritch*, 972 F.2d 1260, 23 USPQ2d 1780 (Fed.Cir. 1992).

[23] 234 F.3d 654, 57 USPQ2d 1161 (Fed.Cir. 2000), citing *In re Dembiczak*, 175 F.3d at 999 (emphasis added here by Appellant).

[24] *In re Thrift*, 298 F.3d 1357 (Fed.Cir. 2002).

[25] *In re Thrift,* 298 F.3d at 1366 (emphasis added).

The Board too has held to this standard; see e.g. *Ex parte Levengood*, 28 USPQ2d 1300, 1304-05 (B.P.A.I. 1993), *Ex parte Obukowicz*, 27 USPQ2d 1063, 1065 (B.P.A.I. 1992), and *Ex parte Clapp*, 227 USPQ 972, 973 (B.P.A.I. 1985).

Even the Manual of Patent Examining Procedure, citing many of these cases, uses an entire subsection (MPEP §§2143.03) to emphasize that an alleged motivation must meet ALL LIMITATIONS of the claimed invention.[26]

Claims 1 and 4

Are Claims 1 and 4 obvious over Peddada et al. in view of Porterfield?

No motivation to modify or combine the references in a way that meets the claimed invention of Claim 1.

Specifically, Claim 1 recites, "**selectively, when commanded by a software application, allowing said accelerator logic to read textures directly from said main memory without downloading them into said graphics memory.**"

Peddada et al. does not disclose or suggest allowing the graphics accelerator itself direct access to the main memory.

Although *Peddada et al.* relates to graphics processing, it does not appear to teach or suggest any innovations with regard to graphics accelerators. *Peddada et al.*, instead, relates a 3D graphics driver that manages a texture cache in the local graphics memory. As correctly noted by Examiner Tung, "*Peddada fails to explicitly suggest or teach, selectively,*

[26] Of course, the MPEP is not authority which binds the Board but is noted merely as a convenient and well-written summary of relevant points of case law.

when commanded by a software application, allowing said accelerator logic to read textures directly from said main memory without downloading them into said graphics memory."[27]

Peddada et al. expressly teaches away from allowing the graphics accelerator direct access to the main memory.

Examiner Tung has suggested that it would have been obvious to combine the AGP model of *Peddada et al.* with the AGP Execute model of *Porterfield* to add flexibility to the system. However, *Peddada et al.* expressly teaches away from the combination suggested by Examiner Tung:

> *Unfortunately, 3D graphics accelerator 20 must have additional hardware to directly access textures from AGP memory 14. This extra hardware adds to the expense and complexity of 3D graphics accelerator 20 and is thus undesirable.*[28]

A prior art reference may be considered to teach away when:

> [A] person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant.[29]

Peddada et al. only mentions the AGP Execute model in the background of the invention in order to show how it improves upon the prior art of the AGP Execute model. As noted by Examiner Tung, "*Peddada uses a different way*

---

[27] Section 2 of Office Action mailed 06/04/23.

[28] Col. 1., *ll.* 58-62.

[29] *In re Gurley*, 27 F.3d 551, 553, 31 USPQ 2d 1130, 1131 (Fed. Cir. 1994).

to achieve the benefit of AGP Execute model."[30] Therefore, Examiner Tung's suggestion that it would have been obvious to combine the two AGP models is not only unfounded but is also contradictory to the teachings of *Peddada et al.* Accordingly, Examiner Tung has failed to establish a proper motivation for the suggested combination.

The asserted combination of references does not support each limitation of Claim 4.

Claim 4 depends directly from Claim 1 and incorporates all the limitations thereof. Claim 4 also includes additional limitations that are not supported by the asserted combination of references. Specifically, Claim 4 recites, "**wherein said accelerator logic is able to read noncontiguous textures directly from said main memory**."

Porterfield does not disclose or suggest an accelerator logic that is capable of reading non-contiguous textures directly from the main memory.

Although Examiner Tung has suggested that *Porterfield* teaches an accelerator logic that is able to read non-contiguous textures directly from said main memory, this is simply incorrect. *Porterfield* states explicitly, "*the 'execute' model requires an address mapping mechanism to map random pages into a single contiguous, physical address space needed by the graphics accelerator 160.*"[31] Clearly, the graphics accelerator disclosed by *Porterfield* is unable to read noncontiguous textures directly from the main memory. Assistance from the host CPU is still required before the *Porterfield* graphics accelerator can read the textures from the main memory. The graphics accelerators of the present application do not require that the textures be physically contiguous and present before they can be used by the graphics accelerator. As a result, no assistance from the host

---

[30] Section 2 of Office Action mailed 06/04/2003.

[31] Col. 7, *ll.* 2-5.

CPU is required. Requiring a mapping mechanism to map random pages into a single contiguous, physical address space before the texture can be used by the graphics accelerator is a problem that is addressed and solved by the present inventions. As stated in the present application:

> **Texture maps can be stored in physical memory or in logic/virtual memory. If the texture map is stored in physical memory then it must be physically contiguous and present before that texture is used.**
>
> - **The management of physical textures is complicated by the fact that an application can request more textures than can fit into on-card memory so the textures need to be dynamically swapped, however this is not an easy task to do ...**
>
> **There are a number of solutions to solving this problem:**
>
> - **Increase the amount of physical memory to hold texture maps. ...**
> - **Allow textures to be executed out of host memory via the AGP or PCI bus. ...**
> - **The final solution is to treat the texture addresses as logical or virtual addresses. The logical part allows the dynamic paging of textures out of host or system memory with or without any assistance from the host CPU. This is done on demand so borrows many of the techniques used for CPU memory management. The virtual texture management (of which the logical addressing is a necessary sub-set) is implemented as standard in this unit and will now be described in detail.**[32] (quoted in full above)

Therefore, *Porterfield* does not even address the problem or the solution disclosed by the present application.

---

[32] Page 31, line 9 – Page 32, line 10.

Therefore, for the reasons discussed above, Appellant respectfully submits that Claims 1 and 4 are not obvious over *Peddada et al.* in view of *Porterfield.*

Claims 2 and 5

Are Claims 2 and 5 obvious over Porterfield in view of Mergard et al. or Poirion or Chen et al?

The asserted combination of references does not support each limitation of Claim 2.

Specifically, Claim 2 recites, "`a graphics accelerator software, integrated on said chip, which has a user accessible mechanism in place to do logical-to-physical mapping into a main system memory.`"

The mapping mechanims disclosed by Porterfield does not appear to be user accessible.

Examiner Tung has suggested that *Porterfield* discloses software that has a user accessible mechanism in place to do logical-to-physical mapping into a main system memory. However, the standard GART mapping mechanism used by *Porterfield* does not appear to be user accessible.

No motivation to modify or combine the references in a way that meets the claimed invention of Claim 2.

As correctly noted by Examiner Tung, *Porterfield* fails to explicitly teach or suggest software, which has a user accessible mechanism in place to do logical-to-physical mapping into a main system memory, integrated on the graphics accelerator chip. Examiner Tung has suggested that it would have been obvious to integrate the software on the graphics accelerator chip in order to achieve a low cost, low space system without sacrificing overall

performance. However, Appellant does not understand how integrating the software on the chip would accomplish this. Examiner Tung cites *Mergard et al.*, *Poirion*, and *Chen et al.* as examples of integrated chips. However, the advantages derived from these chips come from integrating various hardware components onto a single chip, not software. Integrating software on a chip would not give any of the advantages suggested by Examiner Tung. In fact, *Porterfield* expressly teaches away from integrating the software on the chip:

> *[B]y defining the GART in software, the present invention eliminates many hardware dependencies. Instead of expensive circuit redesigns and fabrication, the present invention enables inexpensive software modifications to address future partitioning and remapping circuitry as well as any current for future compatibility issues. Moreover, the present invention enables computer manufacturers to investigate cost and performance compromises at the system integration stage rather than at the hardware design and development stage. ... The invention thus provides substantial flexibility to address ever changing cost and performance requirements well after the completion of the hardware design. In contrast to existing hardware design paradigms, the present invention enables rapid and inexpensive modifications to address evolving customer and market needs.*[33]

Therefore, integrating the GART software on the graphics accelerator chip would result in a loss of the flexibility that was intended by *Porterfield.* Accordingly, even if the GART software used by Porterfield was user accessible (which Appellant strongly disputes), Examiner Tung has failed to establish a proper motivation for the suggested combination.

Claim 5 depends directly from Claim 2 and incorporates all the limitations thereof. Therefore, for this reason and for the reasons discussed

---

[33] Col. 17, *ll.* 12-31.

above, Appellant respectfully submits that Claims 2 and 5 are not obvious over *Porterfield* in view of *Mergard et al.* or *Poirion* or *Chen et al.*

Are Claims 2 and 5 obvious over Peddada et al. in view of Mergard et al. or Poirion or Chen et al?

The asserted combination of references does not support each limitation of Claim 2.

Peddada et al. also fails to explicitly teach or suggest software that has a user accessible mechanism in place to do logical-to-physical mapping into a main system memory.

Examiner Tung has suggested that cache management process disclosed by *Peddada et al.* contains software that has user accessible mechanism in place to do logical-to-physical mapping into a main system memory. However, Appellant is unable to find such a teaching in *Peddada et al.* The section of *Peddada et al.* relied upon by Examiner Tung for this rejection merely states:

> *A cache management process is called by the handle-texture process. It generates a free block in the texture cache. The cache management process returns a free address of the free block to the handle-texture process. The AGP-DMA process uses the free address as a destination address when moving the texture from the main memory. Thus the high-level application calls the set-render process of the graphics driver, which copies the texture in the main memory to the texture cache readable by the 3D graphics engine.*[34]

Logical-to-physical mapping into the main memory is not the same as managing a texture cache in the local graphics memory. Although *Peddada et al.* does mention GART hardware as an alternative embodiment, there is

---

[34] Col. 4, *ll.* 4-12.

no mention of a user accessible software to perform logical-to-physical mapping into a main system memory. Accordingly, Peddada *et al.* does not support this limitation of Claim 2.

No motivation to modify or combine the references in a way that meets the claimed invention of Claim 2.

Again, Examiner Tung has suggested that it would have been obvious to integrate software, which has a user accessible mechanism in place to do logical-to-physical mapping into a main system memory, on the graphics accelerator chip in order to achieve a low cost, low space system without sacrificing overall performance. However, Appellant does not understand how integrating the software on the chip would accomplish this. Examiner Tung cites *Mergard et al.*, *Poirion*, and *Chen et al.* as examples of integrated chips. However, the advantages derived from these chips come from integrating various hardware onto a single chip, not software. Therefore, even if one were to assume that *Peddada et al.* teaches such a software (which Appellant strongly disputes), integrating such a software on a chip would not result in any of the advantages suggested by Examiner Tung. Again, Examiner Tung has failed to establish a proper motivation for the suggested combination.

Claim 5 depends directly from Claim 2 and incorporates all the limitations thereof. Therefore, for this reason and for the reasons discussed above, Appellant respectfully submits that Claims 2 and 5 are not obvious over *Peddada et al.* in view of *Mergard et al.* or *Poirion* or *Chen et al.*

Claims 3 and 6

Are Claims 3 and 6 obvious over Porterfield in view of Mergard et al. or Poirion or Chen et al?

The asserted combination of references does not support each limitation of Claim 3.

Specifically, Claim 3 recites, "`a texture memory management function, integrated on said chip, which manages both texture storage in host memory and also texture storage in normal texture memory`."

Examiner Tung has suggested that managing both texture storage in host memory and texture storage in normal texture memory is a part of the function of the graphics accelerator disclosed by *Porterfield*. However, this is simply incorrect. Although *Porterfield* appears to disclose a graphics accelerator that is capable of accessing both main memory and the local frame buffer, accessing memory is not the same as managing memory. Appellant is unable to find any mention of memory management by the graphics accelerator disclosed in *Porterfield*. Accordingly, *Porterfield* does not support this limitation of Claim 3.

Accessing memory is not the same as managing memory.

Memory management can be viewed as a collection of techniques for providing sufficient memory to one or more processes in a computer system, especially when the system does not have enough memory to satisfy all processes' requirements simultaneously. Techniques include swapping, paging, and virtual memory.[35]

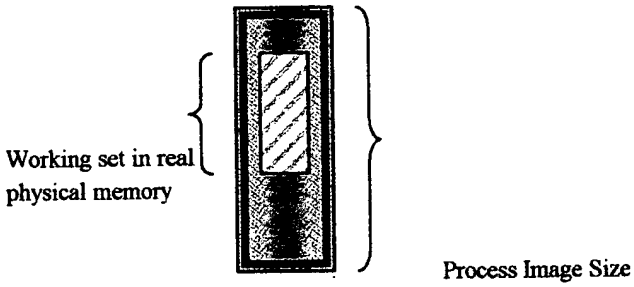
Virtual memory, as employed by the present application, presents a level of indirection between the addresses that an application views, and the physical memory addresses used by the hardware. The benefits of virtual

---

[35] This particular teaching illustrated at http://www.hyperdictionary.com/dictionary/memory+management.

memory include: security, reliability, application transparent relocation of physical memory, and cache partitioning.

Virtual Memory Management loads all process images in parts called "Working-Sets".[36]



Virtual memory uses disk storage space to make the computer work as if it had more memory. When a file or program is too big for the computer to work with in its memory, part of the data is stored on disk. This virtual storage is divided into segments called pages; each page is correlated with a location in physical memory, or RAM. When an address is referenced, the page is swapped into memory; it is sent back to disk when other pages must be called. This allows the program to run as if all the data is in memory.[37]

Therefore, memory management involves more than merely accessing main memory.

Claim 6 depends directly from Claim 3 and incorporates all the limitations thereof. Claim 6 also includes additional limitations that are not shown or suggested by the asserted combination of references. Specifically, Claim 6 recites, "**wherein said texture memory management function is able to read noncontiguous textures directly from said main memory**." As discussed above, the *Porterfield* graphics accelerator chip cited by Examiner Tung does not

---

[36] This particular teaching illustrated at http://iris.nyit.edu/~spatanga/Mem_Mangt_swap.doc.

[37] This particular teaching illustrated at http://www.computeruser.com/resources/dictionary/definition.html?lookup=5891.

contain a memory management function, much less one that is capable of reading non-contiguous textures directly from the main memory.

Therefore, for the reasons discussed above, Appellant respectfully submits that Claims 3 and 6 are not obvious over *Porterfield* in view of *Mergard et al.* or *Poirion* or *Chen et al.*

Are Claims 3 and 6 obvious over Peddada et al. in view of Mergard et al. or Poirion or Chen et al?

The asserted combination of references does not support each limitation of Claim 3.

Specifically, Claim 3 recites, "`a texture memory management function, integrated on said chip, which manages both texture storage in host memory and also texture storage in normal texture memory`."

Examiner Tung has suggested that part of the function of the graphics accelerator disclosed by *Peddada et al.* is to access both memories. However, once again, accessing memory is not the same as managing memory. Although the graphics software driver disclosed by *Peddada et al.* does appear to manage texture storage in the local graphics memory, it does not disclose or suggest managing texture storage in the host memory. Accordingly, *Peddada et al.* does not support this limitation of Claim 3.

Claim 6 depends from directly from Claim 3 and incorporates all the limitations thereof. Therefore, for this reason and for the reasons discussed above, Appellant respectfully submits that Claims 3 and 6 are not obvious over *Peddada et al.* in view of *Mergard et al.* or *Poirion* or *Chen et al.*

Grouping of Claims

The claims on appeal do not stand or fall together, since they contain distinct recitations which are relevant to patentability and to the specific rejections stated. Each claim which is argued separately in the preceding sections of this brief (i.e., 1, 2, 3, 4, 5, and 6) should be considered

separately. Argument: The fact that the claims use different formulations (as detailed above) and/or have been argued separately, shows that, if their patentability is not considered separately, any adverse decision would show that some limitations of some claims, and/or some arguments presented, have been unfairly ignored.

Requested Relief

For the reasons advanced above, Appellant respectfully contends that each claim is patentable. Therefore, reversal of all rejections is respectfully requested.

To the extent necessary, a petition for an extension of time under 37 C.F.R. 1.136 is hereby made. Please charge any shortage in fees due in connection of this paper, including extension of time fees, to Deposit Account 07-2320 and please credit any excess fees to such deposit account.

Respectfully submitted,

N. Elizabeth Pham, Reg.No. 49,042

Customer Number 29106

Attorney for Appellant

11330 Valley Dale Drive, Dallas TX 75230

214-363-3038 *groover@technopatents.com* February 4, 2004

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of:	:
Baldwin	: Art Unit: 2676
AN 09/591,226	: Examiner: Tung, Kee M.
Filed: 06/09/2000	: Atty's Docket: TD-156

For: AUTONOMOUS ADDRESS TRANSLATION IN GRAPHICS SUBSYSTEM (confirmation no. 3469)

APPENDIX A – Text of Claims on Appeal

1. A graphics processing method, comprising the steps of:

(a.) performing 3D-graphics rendering in a graphics accelerator subsystem, using a dedicated graphics memory as primary memory for rendering accelerator logic;

(b.) using a system main memory as additional memory to hold textures required by said rendering accelerator logic; and

(c.) when textures required by said rendering accelerator logic are not present in said dedicated graphics memory, then either

downloading said textures from main memory into said graphics memory, or

selectively, when commanded by a software application, allowing said accelerator logic to read textures directly from said main memory without downloading them into said graphics memory.

2. A graphics processing chip, comprising:

a graphics accelerator comprising rendering acceleration logic; and

software, integrated on said chip, which has a user accessible mechanism in place to do logical-to-physical mapping into a main system memory.

3. A graphics processing chip, comprising:

a graphics accelerator comprising rendering acceleration logic; and

a texture memory management function, integrated on said chip, which manages both texture storage in host memory and also texture storage in normal texture memory.

4. The method of Claim 1, wherein said accelerator logic is able to read noncontiguous textures directly from said main memory.

5. The chip of Claim 2, wherein said software is able to read noncontiguous textures directly from said main memory.

6. The chip of Claim 3, wherein said texture memory management function is able to read noncontiguous textures directly from said main memory.

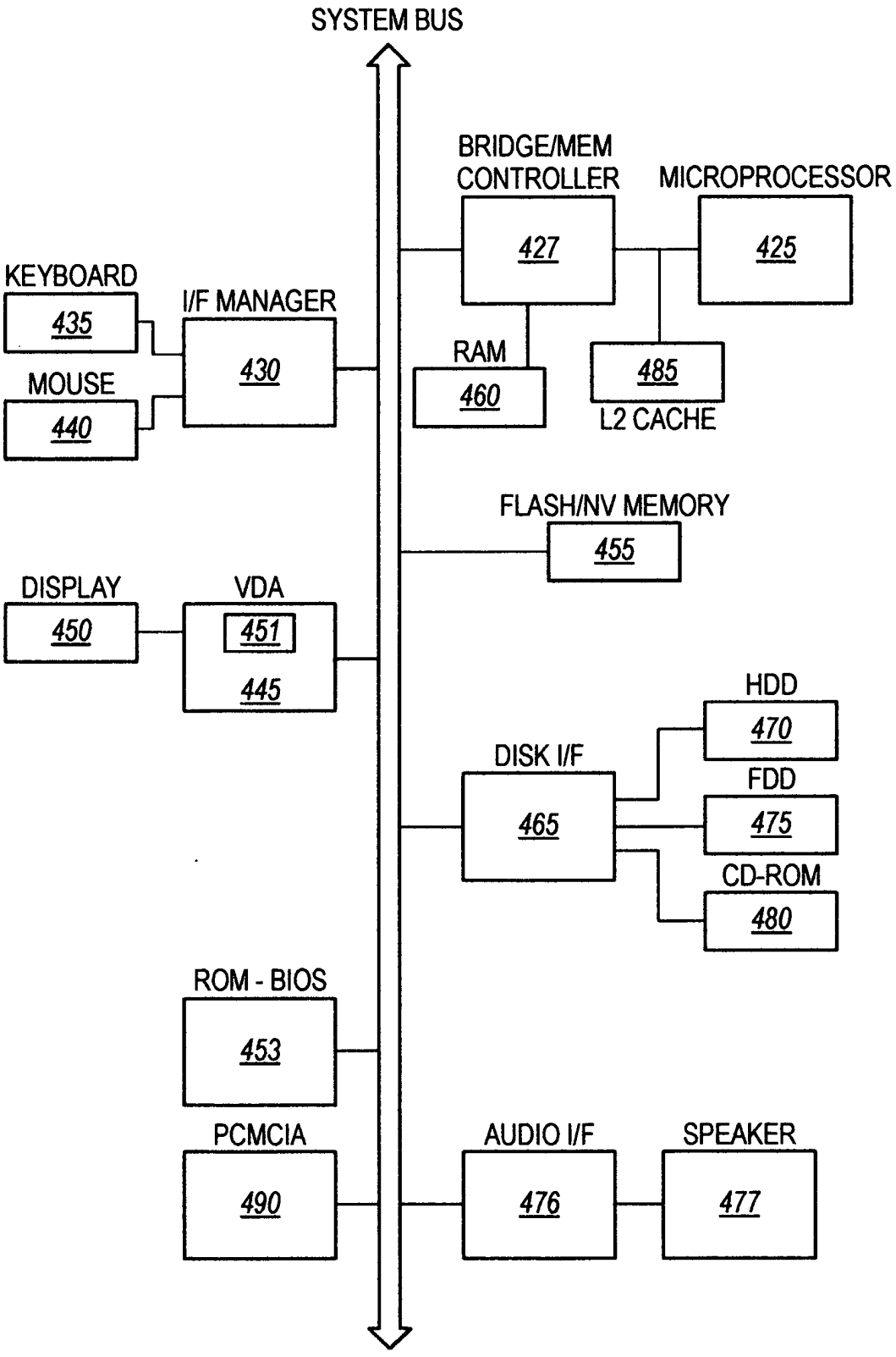

SYSTEM BUS
BRIDGE/MEM CONTROLLER
427
MICROPROCESSOR
425
KEYBOARD
435
I/F MANAGER
430
MOUSE
440
RAM
460
485
L2 CACHE
FLASH/NV MEMORY
455
DISPLAY
450
VDA
451
445
HDD
470
DISK I/F
465
FDD
475
CD-ROM
480
ROM - BIOS
453
PCMCIA
490
AUDIO I/F
476
SPEAKER
477

FIG. 1

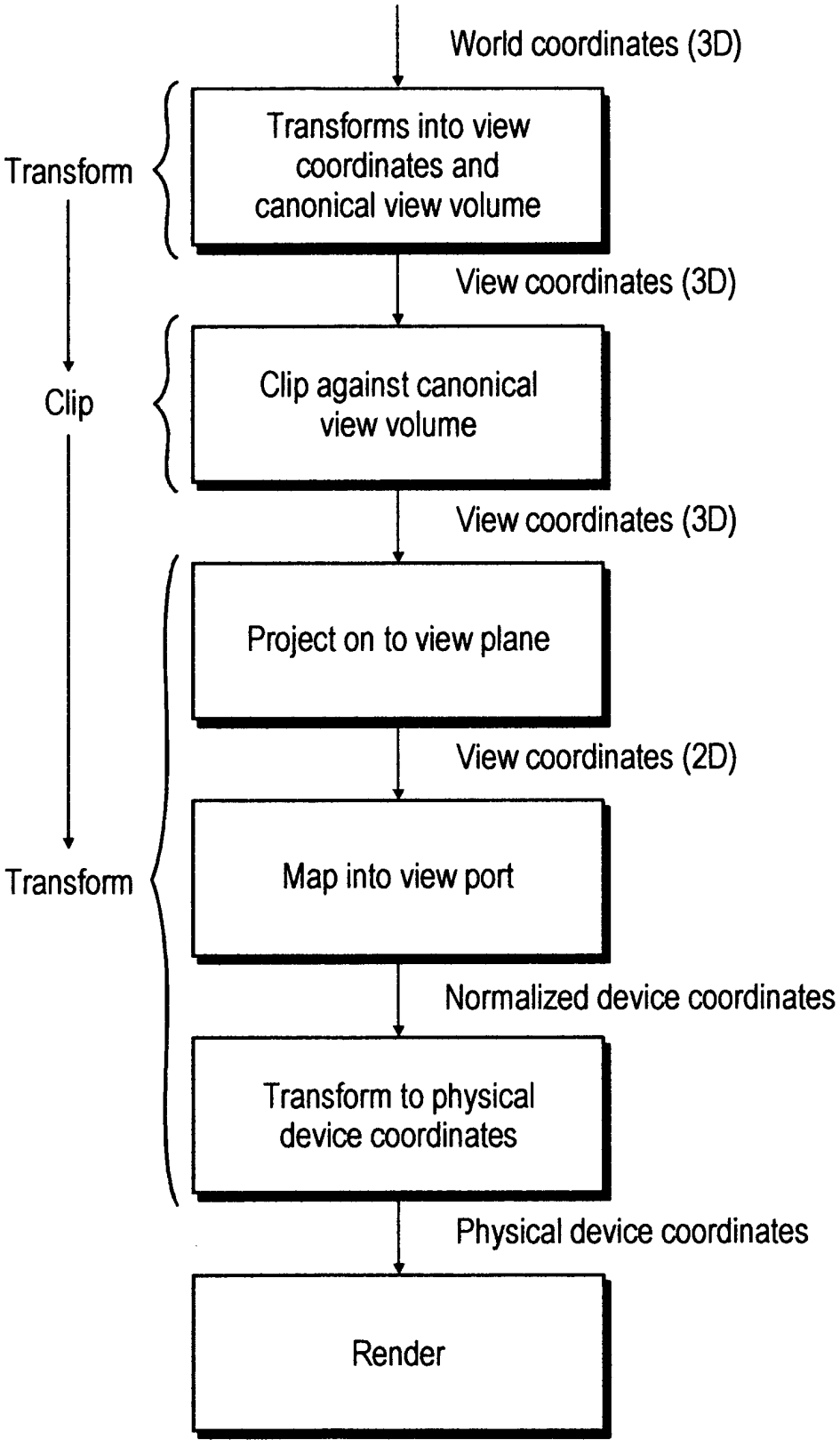

World coordinates (3D)
Transform
Transforms into view coordinates and canonical view volume
View coordinates (3D)
Clip
Clip against canonical view volume
View coordinates (3D)
Project on to view plane
View coordinates (2D)
Transform
Map into view port
Normalized device coordinates
Transform to physical device coordinates
Physical device coordinates
Render

FIG. 2

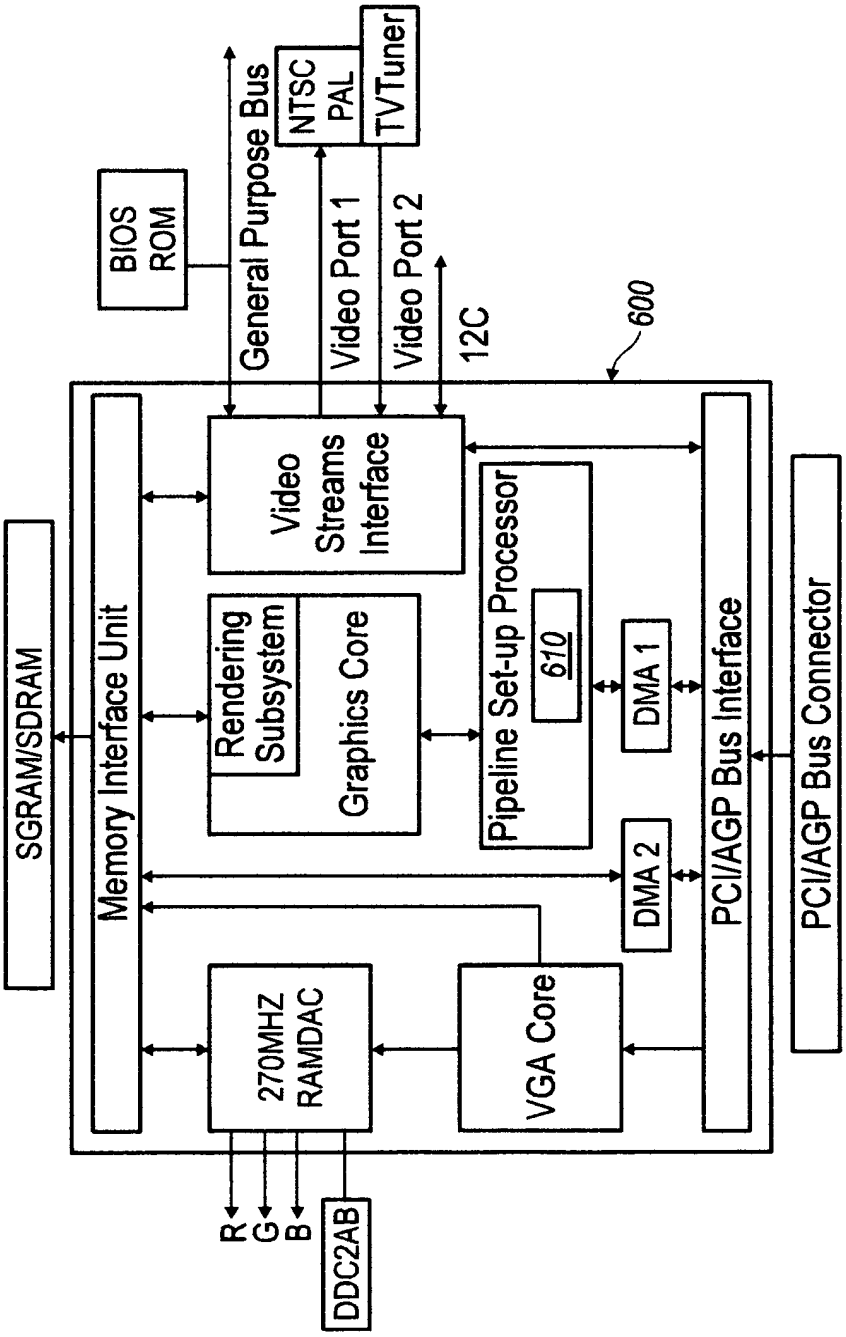

BIOS ROM
General Purpose Bus
NTSC PAL
TVTuner
Video Port 1
Video Port 2
12C
600
Video Streams Interface
Rendering Subsystem
Graphics Core
Pipeline Set-up Processor
610
DMA 1
DMA 2
PCI/AGP Bus Interface
PCI/AGP Bus Connector
SGRAM/SDRAM
Memory Interface Unit
270MHZ RAMDAC
VGA Core
R
G
B
DDC2AB

FIG. 3

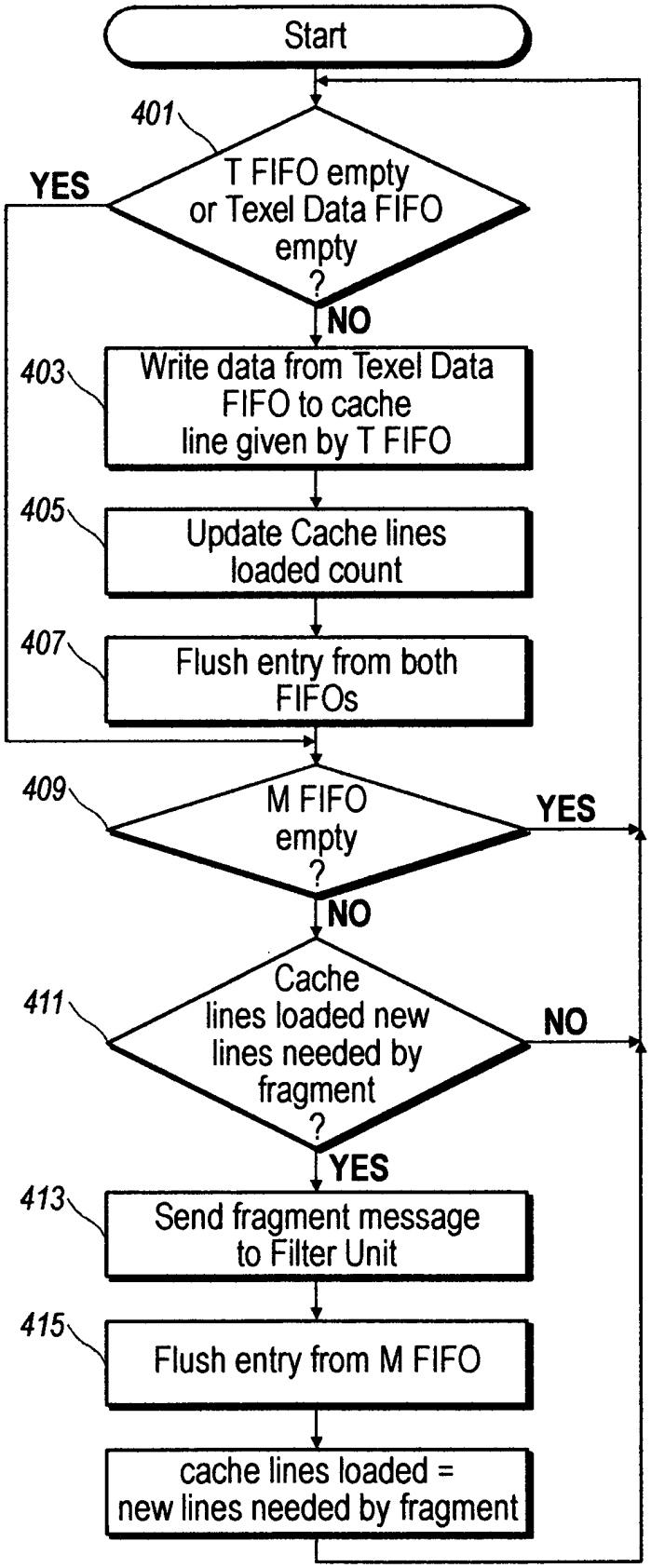

Start
401
T FIFO empty or Texel Data FIFO empty ?
YES
NO
403
Write data from Texel Data FIFO to cache line given by T FIFO
405
Update Cache lines loaded count
407
Flush entry from both FIFOs
409
M FIFO empty ?
YES
NO
411
Cache lines loaded new lines needed by fragment ?
NO
YES
413
Send fragment message to Filter Unit
415
Flush entry from M FIFO
cache lines loaded = new lines needed by fragment

FIG. 4A

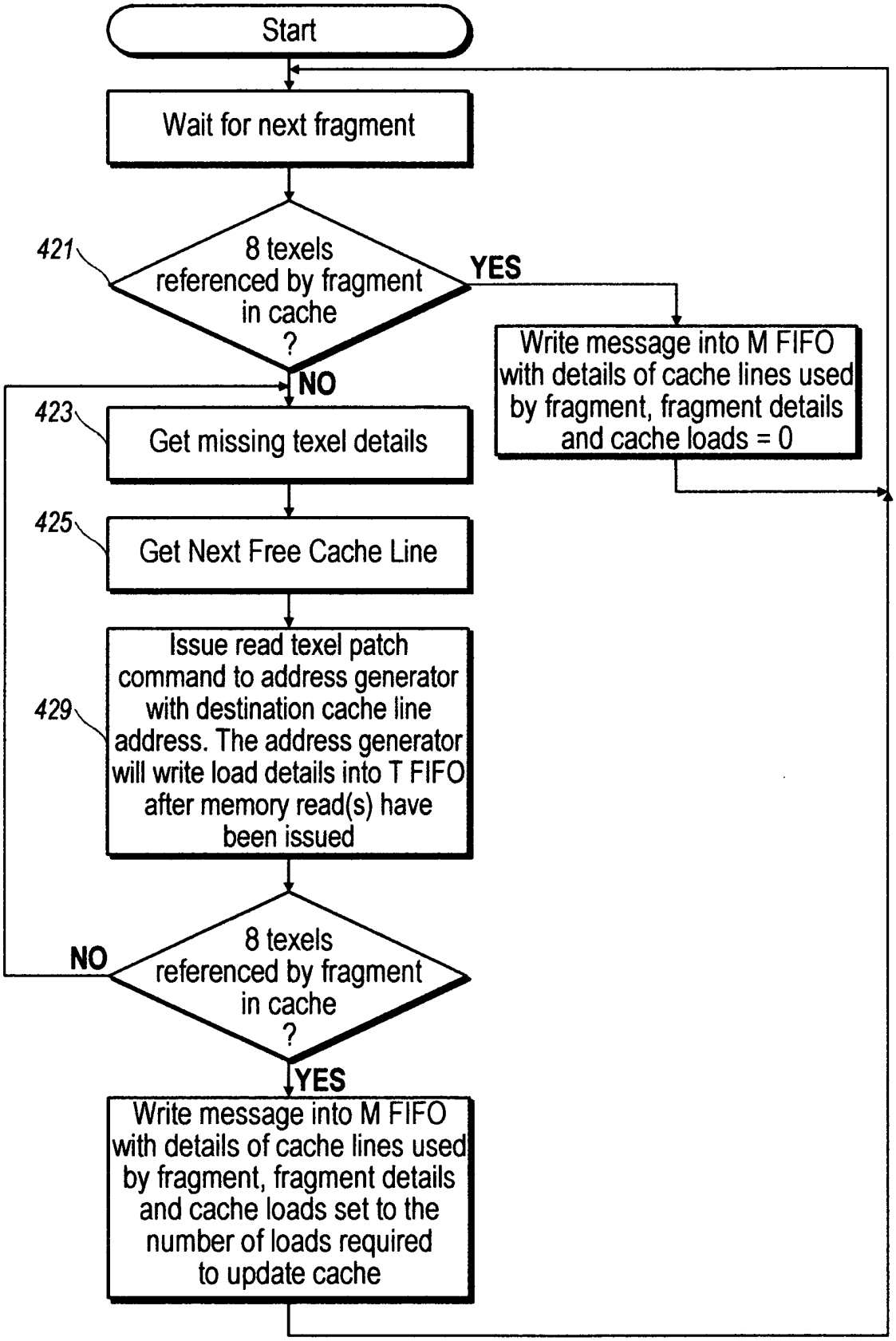

Start
Wait for next fragment
421
8 texels referenced by fragment in cache ?
YES
NO
Write message into M FIFO with details of cache lines used by fragment, fragment details and cache loads = 0
423
Get missing texel details
425
Get Next Free Cache Line
429
Issue read texel patch command to address generator with destination cache line address. The address generator will write load details into T FIFO after memory read(s) have been issued
8 texels referenced by fragment in cache ?
NO
YES
Write message into M FIFO with details of cache lines used by fragment, fragment details and cache loads set to the number of loads required to update cache

FIG. 4B

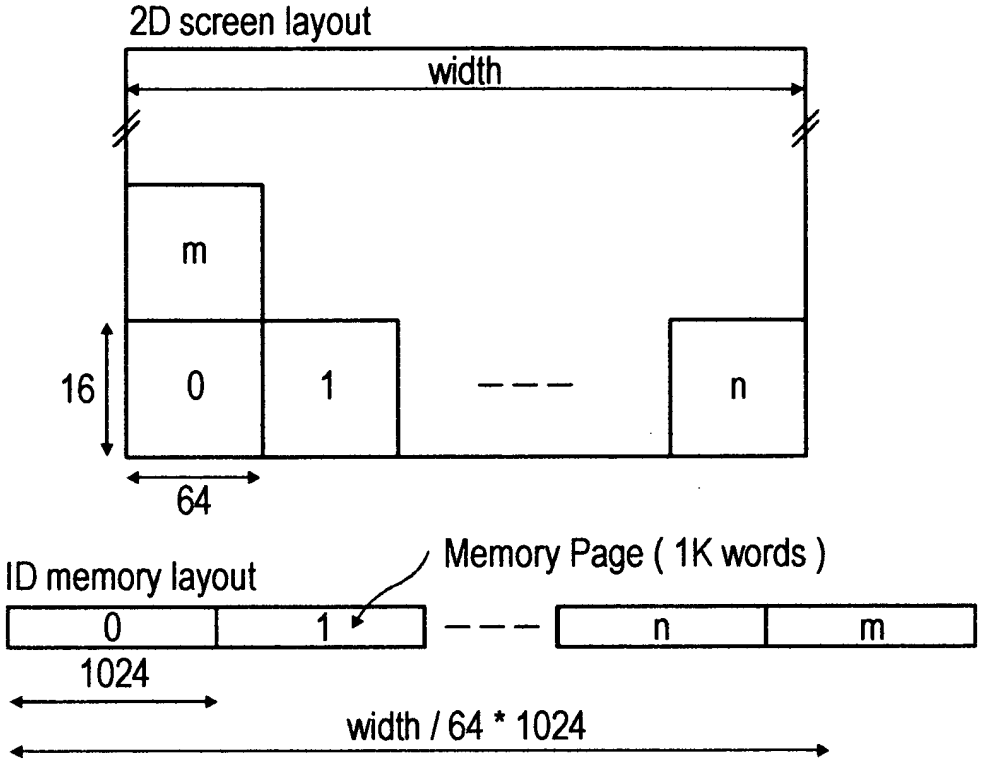

2D screen layout
width
m
16
0
1
---
n
64
ID memory layout
Memory Page (1K words)
0
1
---
n
m
1024
width / 64 * 1024

FIG. 5

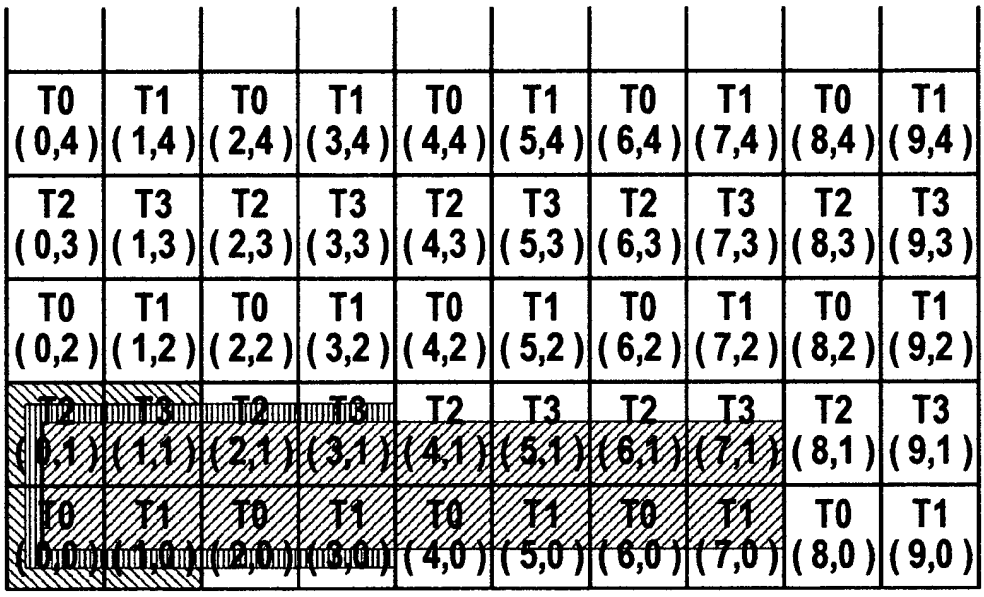
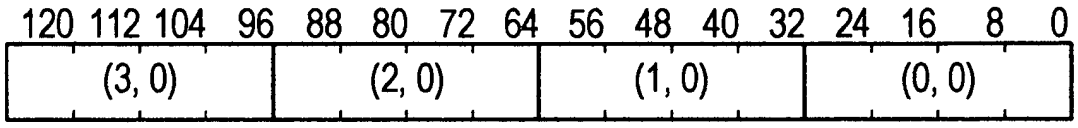

T0 (0,4) T1 (1,4) T0 (2,4) T1 (3,4) T0 (4,4) T1 (5,4) T0 (6,4) T1 (7,4) T0 (8,4) T1 (9,4)
T2 (0,3) T3 (1,3) T2 (2,3) T3 (3,3) T2 (4,3) T3 (5,3) T2 (6,3) T3 (7,3) T2 (8,3) T3 (9,3)
T0 (0,2) T1 (1,2) T0 (2,2) T1 (3,2) T0 (4,2) T1 (5,2) T0 (6,2) T1 (7,2) T0 (8,2) T1 (9,2)
T2 (0,1) T3 (1,1) T2 (2,1) T3 (3,1) T2 (4,1) T3 (5,1) T2 (6,1) T3 (7,1) T2 (8,1) T3 (9,1)
T0 (0,0) T1 (1,0) T0 (2,0) T1 (3,0) T0 (4,0) T1 (5,0) T0 (6,0) T1 (7,0) T0 (8,0) T1 (9,0)
32 bit texels in memory word
16 bit texels in memory word
8 bit texels in memory word

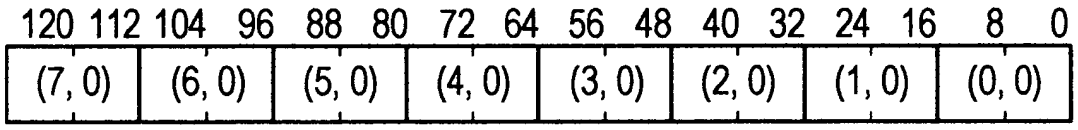
FIG. 6

Linear or Patch64 Memory Layouts

32 bits per texel



16 bits per texel



8 bits per texel

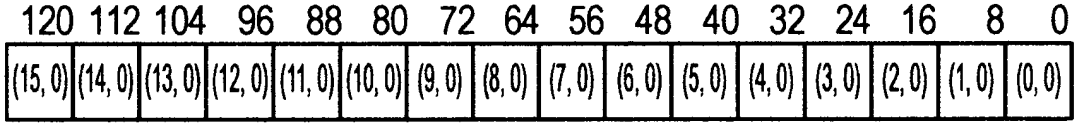
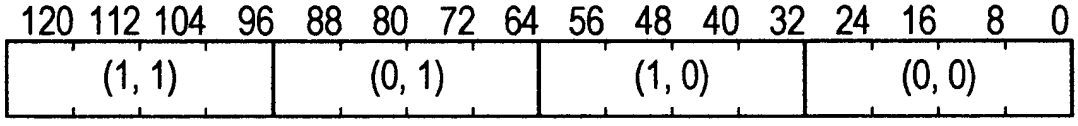


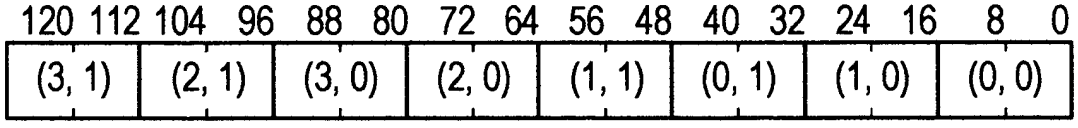
FIG. 7A

Patch32_2 or Patch2 Memory Layouts

32 bits per texel



16 bits per texel



8 bits per texel

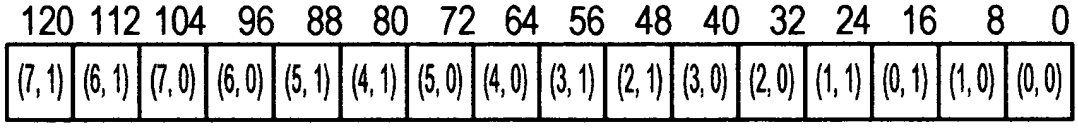


FIG. 7B

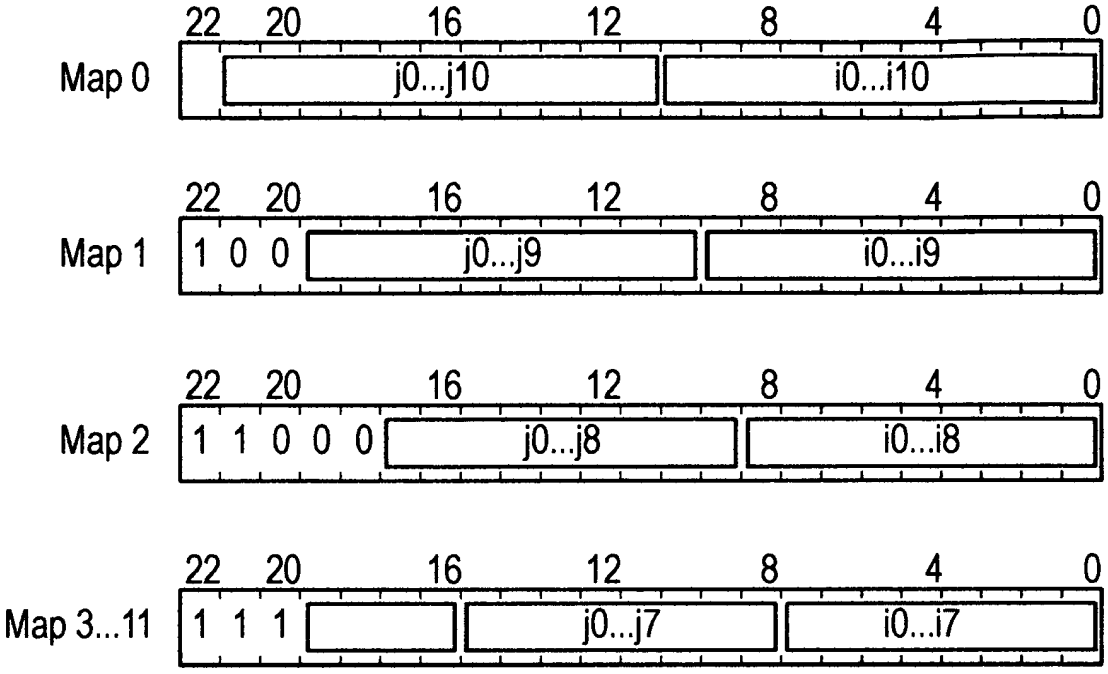

22 20 16 12 8 4 0
Map 0
j0...j10
i0...i10
22 20 16 12 8 4 0
Map 1
1 0 0
j0...j9
i0...i9
22 20 16 12 8 4 0
Map 2
1 1 0 0 0
j0...j8
i0...i8
22 20 16 12 8 4 0
Map 3...11
1 1 1
j0...j7
i0...i7

FIG. 8

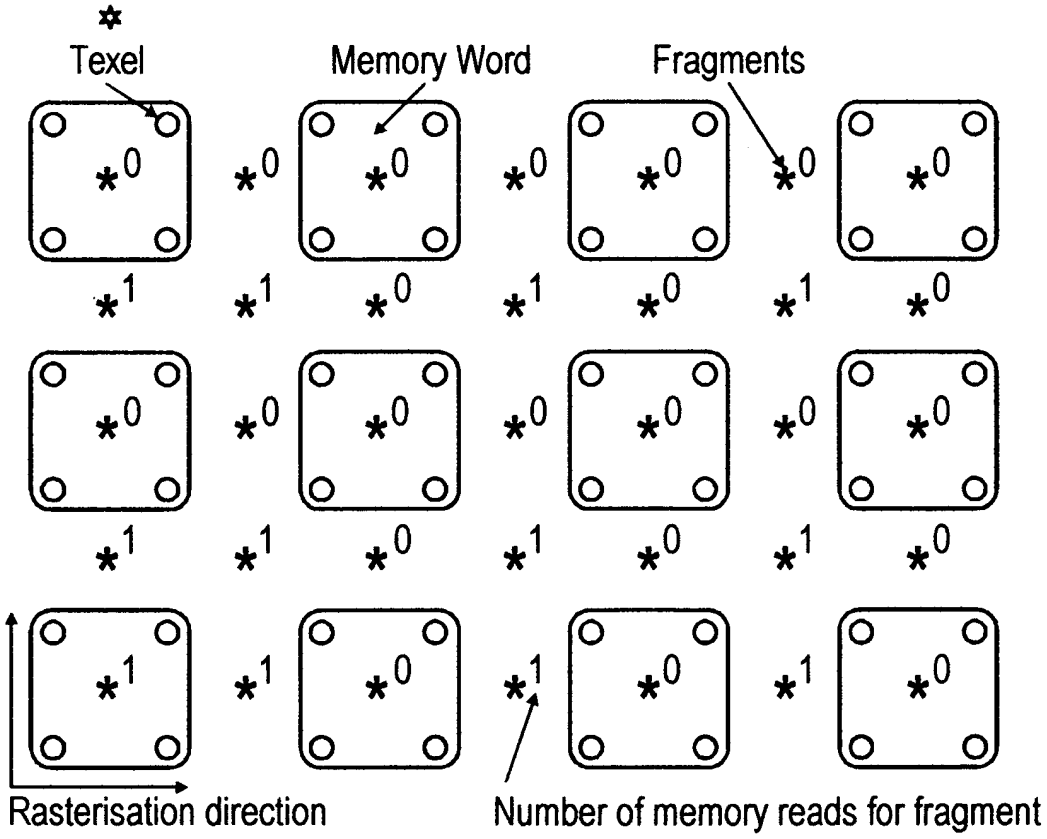

Texel
Memory Word
Fragments
Rasterisation direction
Number of memory reads for fragment

FIG. 9

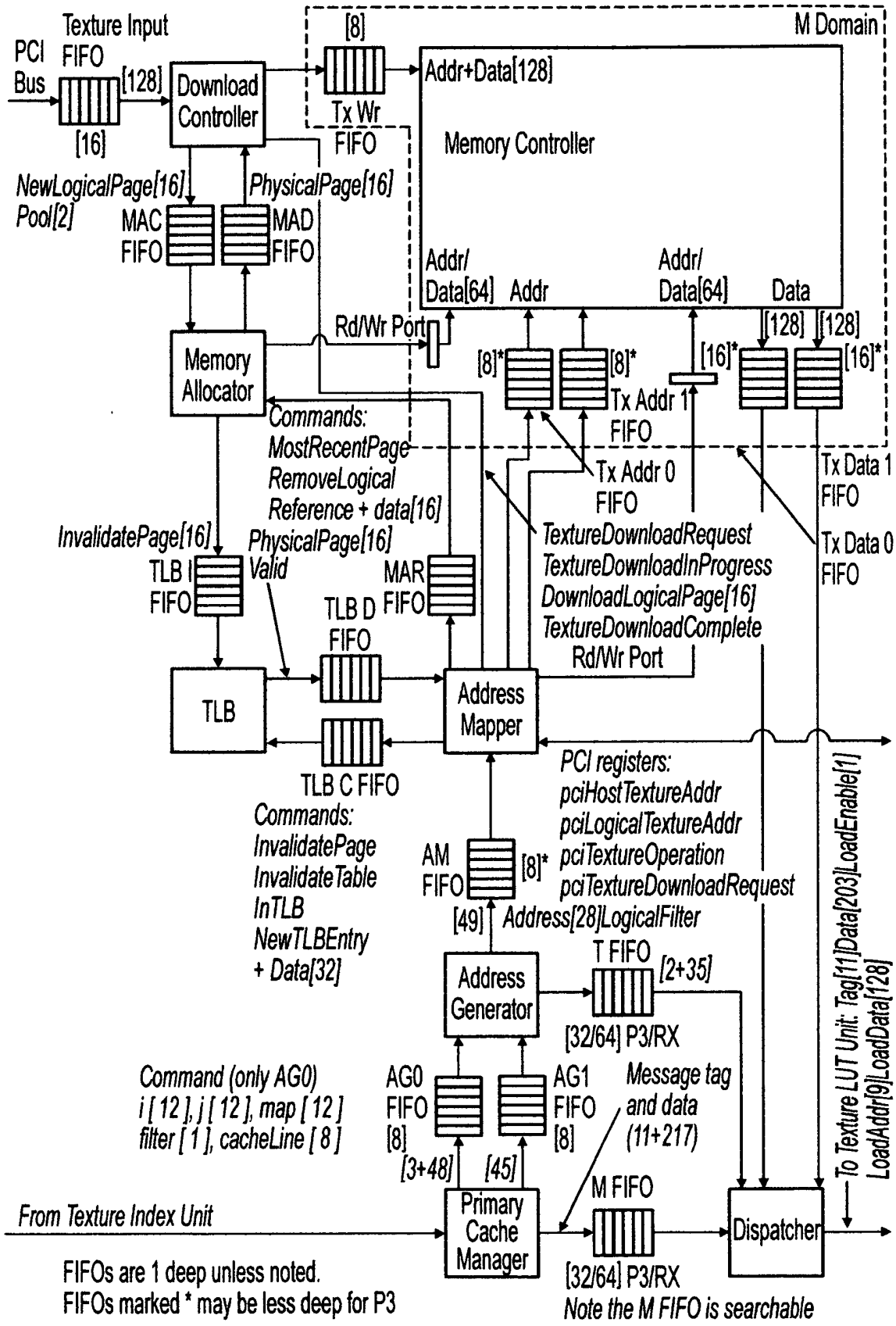

Texture Input
PCI Bus
FIFO
[128]
[16]
Download Controller
[8]
Tx Wr FIFO
M Domain
Addr+Data[128]
Memory Controller
NewLogicalPage[16]
Pool[2]
MAC FIFO
MAD FIFO
PhysicalPage[16]
Addr/ Data[64]
Addr
Addr/ Data[64]
Data
Rd/Wr Port
[8]*
[16]*
[128]
Memory Allocator
Commands:
MostRecentPage
RemoveLogical
Reference + data[16]
Tx Addr 1 FIFO
Tx Addr 0 FIFO
Tx Data 1 FIFO
Tx Data 0 FIFO
InvalidatePage[16]
PhysicalPage[16]
Valid
TLB I FIFO
MAR FIFO
TLB D FIFO
TextureDownloadRequest
TextureDownloadInProgress
DownloadLogicalPage[16]
TextureDownloadComplete
Rd/Wr Port
TLB
Address Mapper
TLB C FIFO
PCI registers:
pciHostTextureAddr
pciLogicalTextureAddr
pciTextureOperation
pciTextureDownloadRequest
Commands:
InvalidatePage
InvalidateTable
InTLB
NewTLBEntry
+ Data[32]
AM FIFO
[8]*
[49]
Address[28]LogicalFilter
Address Generator
T FIFO
[2+35]
[32/64] P3/RX
Command (only AG0)
i [12], j [12], map [12]
filter [1], cacheLine [8]
AG0 FIFO [8]
AG1 FIFO [8]
Message tag and data (11+217)
[3+48]
[45]
M FIFO
From Texture Index Unit
Primary Cache Manager
Dispatcher
To Texture LUT Unit: Tag[11]Data[203]LoadEnable[1] LoadAddr[9]LoadData[128]
[32/64] P3/RX
FIFOs are 1 deep unless noted.
FIFOs marked * may be less deep for P3
Note the M FIFO is searchable

FIG. 10

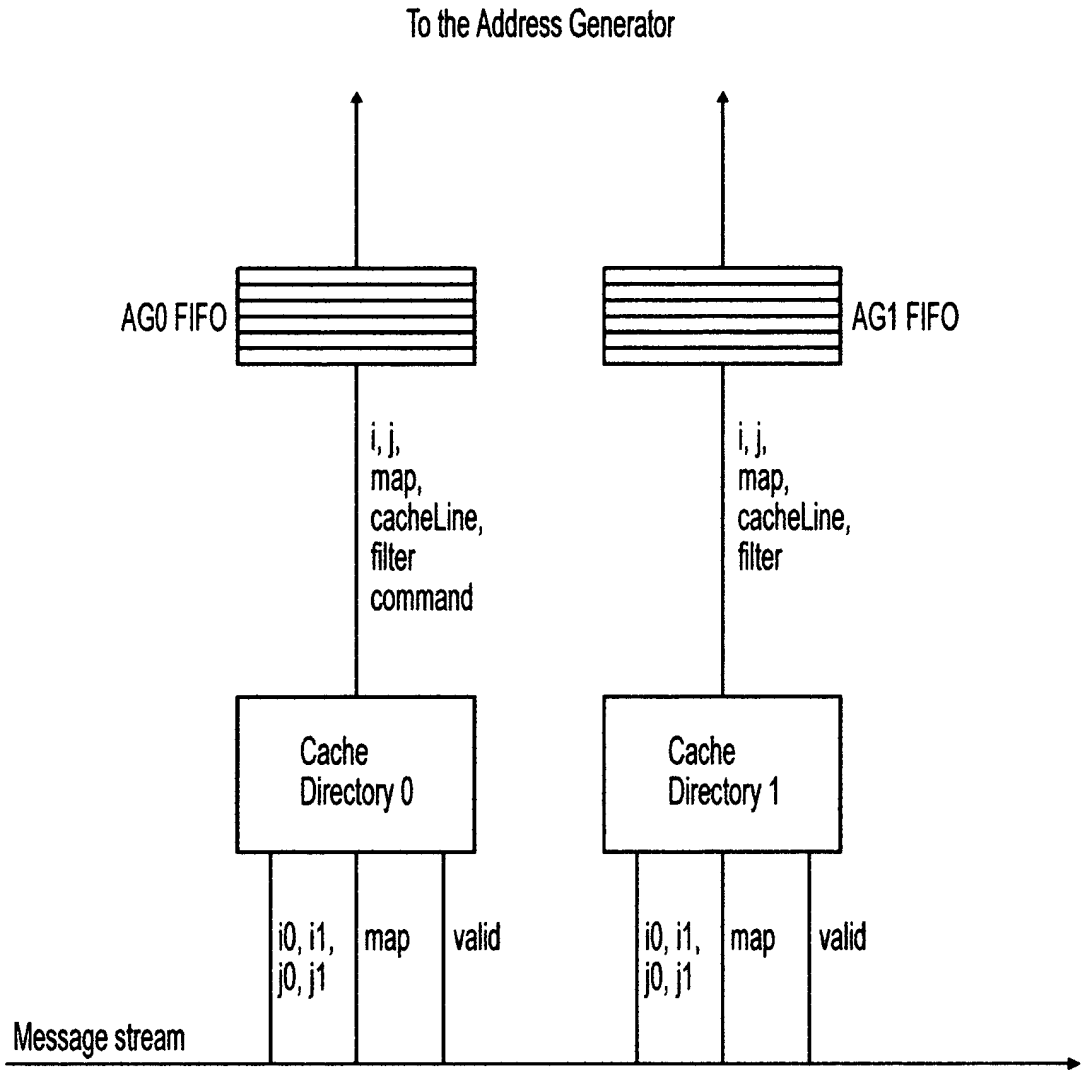

To the Address Generator
AG0 FIFO
AG1 FIFO
i, j,
map,
cacheLine,
filter
command
i, j,
map,
cacheLine,
filter
Cache
Directory 0
Cache
Directory 1
i0, i1,
j0, j1
map
valid
i0, i1,
j0, j1
map
valid
Message stream

FIG. 11

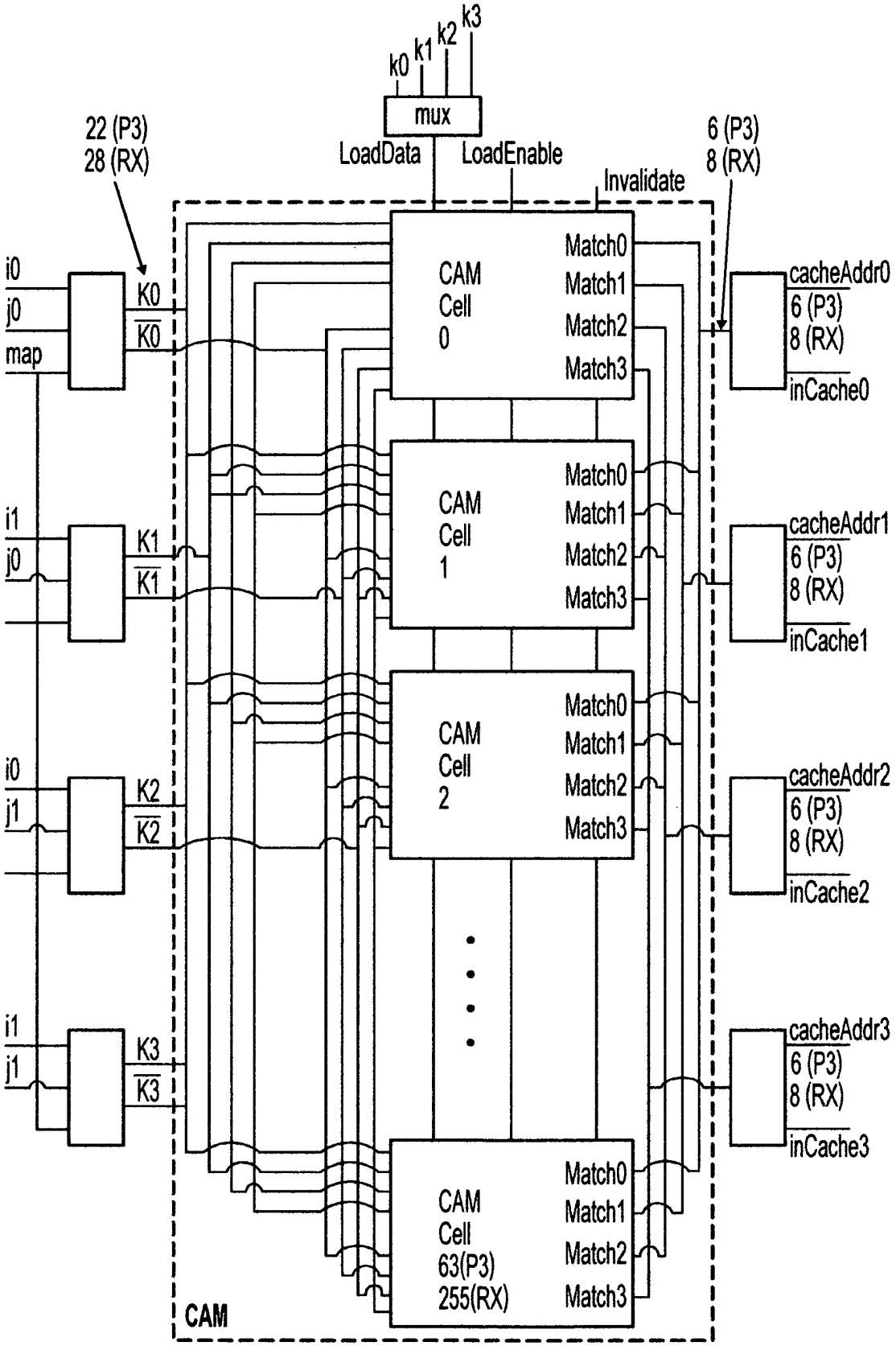

k0
k1
k2
k3
mux
22 (P3)
28 (RX)
LoadData
LoadEnable
Invalidate
6 (P3)
8 (RX)
i0
j0
map
K0
$\overline{K0}$
i1
j0
K1
$\overline{K1}$
i0
j1
K2
$\overline{K2}$
i1
j1
K3
$\overline{K3}$
CAM
Cell
0
CAM
Cell
1
CAM
Cell
2
CAM
Cell
63(P3)
255(RX)
Match0
Match1
Match2
Match3
cacheAddr0
6 (P3)
8 (RX)
inCache0
cacheAddr1
6 (P3)
8 (RX)
inCache1
cacheAddr2
6 (P3)
8 (RX)
inCache2
cacheAddr3
6 (P3)
8 (RX)
inCache3
CAM

FIG. 12

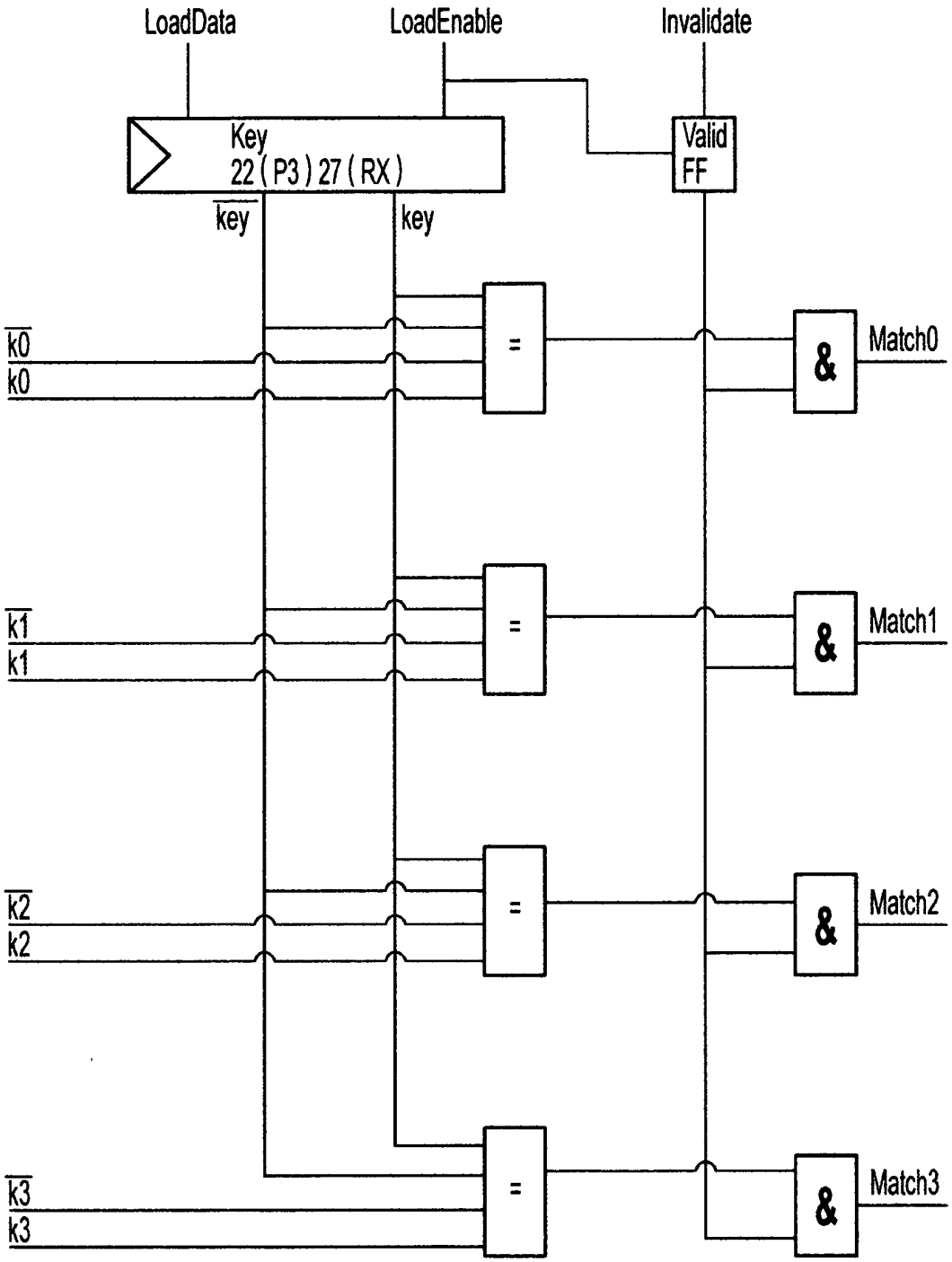

LoadData
LoadEnable
Invalidate
Key
22 (P3) 27 (RX)
Valid
FF
key
key
k0
k0
=
&
Match0
k1
k1
=
&
Match1
k2
k2
=
&
Match2
k3
k3
=
&
Match3

FIG. 13

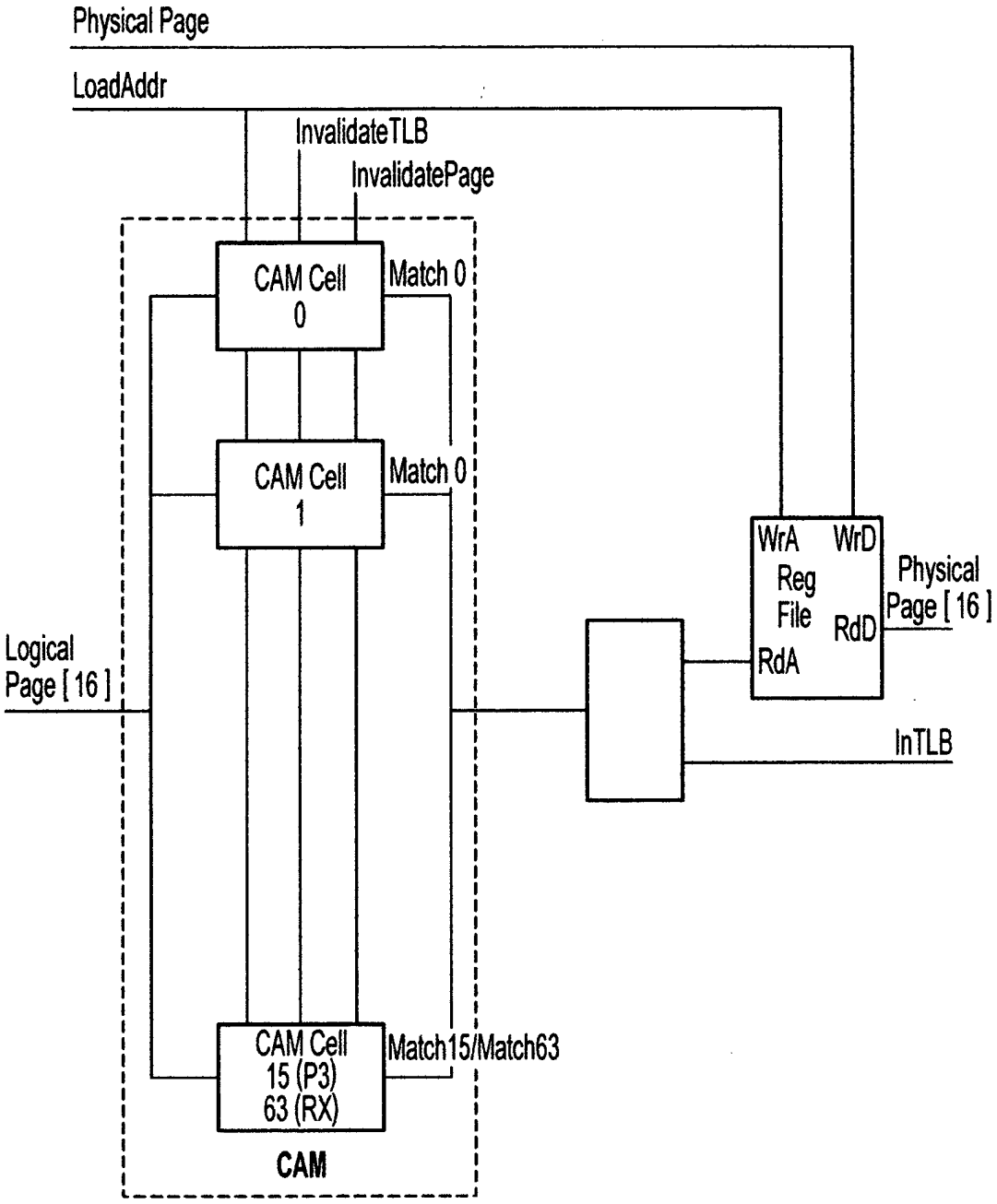

Physical Page
LoadAddr
InvalidateTLB
InvalidatePage
CAM Cell 0
Match 0
CAM Cell 1
Match 0
CAM Cell 15 (P3) 63 (RX)
Match15/Match63
CAM
WrA
WrD
Reg File
RdD
RdA
Physical Page [16]
Logical Page [16]
InTLB

FIG. 14

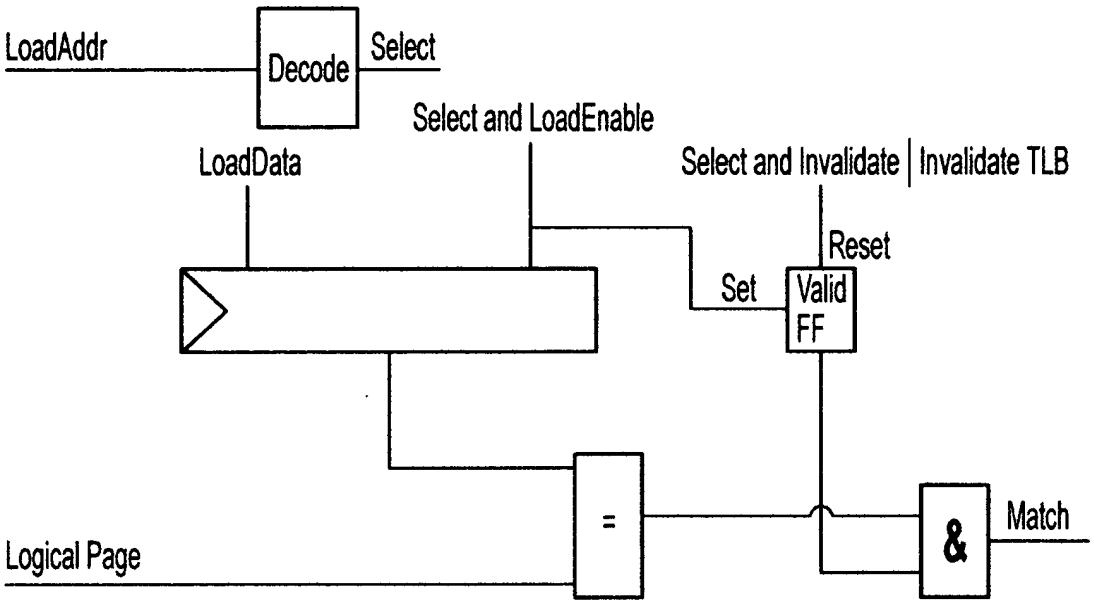

LoadAddr
Decode
Select
Select and LoadEnable
LoadData
Select and Invalidate
Invalidate TLB
Reset
Set
Valid FF
=
&
Match
Logical Page

FIG. 15

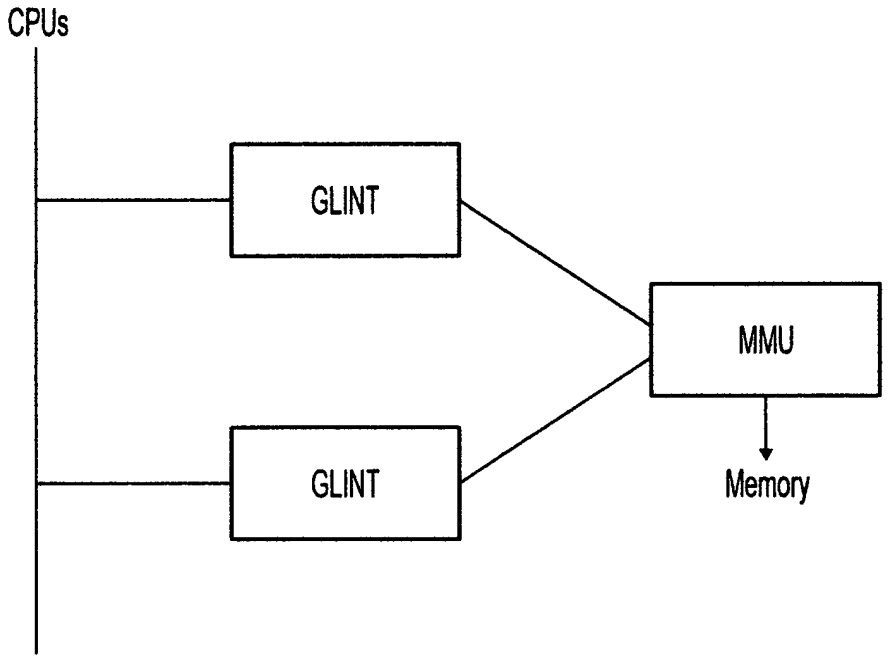

CPUs
GLINT
GLINT
MMU
Memory

FIG. 16

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re application of: :
Baldwin : Art Unit: 2676
AN 09/591,226 : Examiner: Tung, Kee M.
Filed: 06/09/2000 : Atty's Docket: TD-156
For: Autonomous Address Translation In Graphics Subsystem (confirmation no. 3469)

APPEAL BRIEF

Honorable Commissioner of Patents and Trademarks
Alexandria, VA 22313

Sir:

Enclosed are four complete copies (three bound and one unbound) of the appeal brief submitted in support of the above application.

Any extension of time necessary to prevent abandonment has been requested, and any fee necessary for consideration of this paper has been authorized to be charged to Deposit Account Number 07-2320.

Respectfully submitted,

N. Elizabeth Pham, Reg.No. 49,042

Customer Number 29106

Attorney for Applicant
11330 Valley Dale Drive, Dallas TX 75230
214-363-3038 *groover@technopatents.com* February 4, 2004

AF 2676
$ 2700

PTO/SB/21 (08-03)
Approved for use through 08/30/2003. OMB 0651-0031
U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE
Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it displays a valid OMB control number.

TRANSMITTAL FORM

(to be used for all correspondence after initial filing)

Application Number	09/591,226
Filing Date	06/09/2000
First Named Inventor	Baldwin
Art Unit	2676
Examiner Name	Tung, Kee M.
Attorney Docket Number	TD-156

RECEIVED
FEB 11 2004
Technology Center 2600

Total Number of Pages in This Submission

ENCLOSURES *(Check all that apply)*

- [x] Fee Transmittal Form
- [ ] Fee Attached
- [ ] Amendment/Reply
- [ ] After Final
- [ ] Affidavits/declaration(s)
- [ ] Extension of Time Request
- [ ] Express Abandonment Request
- [ ] Information Disclosure Statement
- [ ] Certified Copy of Priority Document(s)
- [ ] Response to Missing Parts/ Incomplete Application
- [ ] Response to Missing Parts under 37 CFR 1.52 or 1.53
- [ ] Drawing(s)
- [ ] Licensing-related Papers
- [ ] Petition
- [ ] Petition to Convert to a Provisional Application
- [ ] Power of Attorney, Revocation Change of Correspondence Address
- [ ] Terminal Disclaimer
- [ ] Request for Refund
- [ ] CD, Number of CD(s) _____
- [ ] After Allowance communication to Technology Center (TC)
- [x] Appeal Communication to Board of Appeals and Interferences
- [x] Appeal Communication to TC **(Appeal Notice, Brief, Reply Brief)**
- [ ] Proprietary Information
- [ ] Status Letter
- [ ] Other Enclosure(s) (please Identify below):

Remarks

SIGNATURE OF APPLICANT, ATTORNEY, OR AGENT

Firm or Individual name	N. Elizabeth Pham
Signature	NEPham
Date	02/04/2004

CERTIFICATE OF TRANSMISSION/MAILING

I hereby certify that this correspondence is being facsimile transmitted to the USPTO or deposited with the United States Postal Service with sufficient postage as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 on the date shown below.

Typed or printed name	N. Elizabeth Pham		
Signature	NEPham	Date	02/04/2004

This collection of information is required by 37 CFR 1.5. The information is required to obtain or retain a benefit by the public which is to file (and by the USPTO to process) an application. Confidentiality is governed by 35 U.S.C. 122 and 37 CFR 1.14. This collection is estimated to 12 minutes to complete, including gathering, preparing, and submitting the completed application form to the USPTO. Time will vary depending upon the individual case. Any comments on the amount of time you require to complete this form and/or suggestions for reducing this burden, should be sent to the Chief Information Officer, U.S. Patent and Trademark Office, U.S. Department of Commerce, P.O. Box 1450, Alexandria, VA 22313-1450. DO NOT SEND FEES OR COMPLETED FORMS TO THIS ADDRESS. **SEND TO: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.**

If you need assistance in completing the form, call 1-800-PTO-9199 and select option 2.